Visual Search Automation for Unmanned Aerial Vehicles

Eric N. Johnson, Alison A. Proctor, Jincheol Ha, and Allen R. Tannenbaum

Abstract— **This paper describes the design, development, and testing of an Unmanned Aerial Vehicle (UAV) with automated capabilities: searching a prescribed area, identifying a specific building within that area based on a small sign located on one wall, and then identifying an opening into that building. This includes a description of the automated search system along with simulation and flight test results. Results include successful evaluation at the McKenna Military Operations in Urban Terrain flight test site.**

Index Terms— **Image Processing, Automatic Target Recognition, Flight Control, Unmanned Aerial Vehicles.**

I. INTRODUCTION

This paper describes the design, development, and testing of an Unmanned Aerial Vehicle (UAV) system with automated search capabilities. The automated search capabilities allow the system to search a prescribed area, identify a specific building within that search area based on a small identifying sign located on one wall, and identify a candidate opening into that specified building. Subsequent selection of the search area, all functions are automated, and do not require human operator assistance. The applications of these capabilities include reduction of operator workload in operational UAV systems, new UAV or guided-munition missions conducted without the assistance or availability of human operators, or the enhancement/augmentation of human search capabilities.

First, the design reference mission is described, based on the Association for Unmanned Vehicle Systems International (AUVSI) Aerial Robotics Competition (an intercollegiate engineering challenge)[1]. The overall approach to accomplish this mission is outlined. The research UAV system utilized for this work is then described in summary form. Following this, the automated search capability added to this system is described, along with ground simulation and flight test results. Finally, conclusions are drawn based on these efforts.

A. Mission Definition

The design reference mission provides a focus for the work described herein, including specifications for an automated visual search system with real-world applications. This mission is the Level 2 mission of the International Aerial Robotics Competition, rules as specified for 2001-2003 [1]. This is an intercollegiate engineering competition, sponsored by AUVSI, in which university teams build automated UAV systems that perform difficult missions; missions that are selected to push the state-of-the-art in UAV automation. At the time this article was written, Georgia Tech was the only team to have attempted this portion of the challenge; the approach that some of the other teams have taken can be found in [2],[3].

The mission specifies a search area containing any number of buildings and other obstacles. The UAV system must automatically (without human operator control):

1. Fly into the search area.
2. Locate a specific building identified by a 1-meter square sign located on one wall, Fig. 1.
3. Identify an opening into this specific building. The opening can be a window or a door, but it must be open.

Fig. 1. Symbol for building identification located on one wall, actual size is one-meter diameter.

Manuscript received November 7, 2003. This work was supported in part by the DARPA Software Enabled Control (SEC) Program under contracts #33615-98-C-1341 and #33615-99-C-1500, Lockheed Martin, NovAtel, Bahr Avionics, and Inflight Robotics. Other direct contributors include: Henrik Christophersen, J. Eric Corban, Joerg Dittrich, Jeong Hur, Suresh Kannan, Sumit Mishra, Wayne Pickell, Daniel Schrage, and Hungsun Son. Operations at the McKenna MOUT site required the support of McKenna personnel and AUVSI.

E. N. Johnson is with the Georgia Institute of Technology, Atlanta, GA 30332-0150 USA (email: eric.johnson@aerospace.gatech.edu).

A. A. Proctor is with the Georgia Institute of Technology, Atlanta, GA 30332-0150 USA. (404-385-2850 email: alison_proctor@ae.gatech.edu).

J. Ha is with the Georgia Institute of Technology, Atlanta, GA 30332-0150 USA (email: gtg203c@mail.gatech.edu).

A. Tannenbaum with the Georgia Institute of Technology, Atlanta, GA 30332-0150 USA (email: tannenba@bme.gatech.edu)

For 2003, the competition was held at the McKenna Military Operations in Urban Terrain (MOUT) site in Fort Benning, Georgia. This location includes a simulated village with 15 buildings, including normal obstacles such as overhead wires, towers, and trees. The pattern of building locations is irregular, and many of the buildings have unique shapes, window/door configurations, and colors. The pre-specified search area includes the entire village, and the symbol in Fig. 1 can be placed on the wall of any building. An overhead view of the village is shown in Fig. 2.

Fig. 2. Overhead view of the McKenna MOUT site, a simulated village with 15 buildings and other obstacles such as overhead wires and trees; a the white car located at the top of the image near the center provides a sense of scale

B. Approach

A research UAV with automated guidance, navigation, and flight control capabilities was augmented with a new automated visual search subsystem described in this paper. The research UAV utilized is the Georgia Tech Yamaha R-Max based system, referred to as the GTMax [3]. The resulting system has been tested in simulation and flight tests, and was utilized as an entry into the Aerial Robotics Competition in 2003 [1].

The McKenna MOUT site is a well-documented area, and the geographic coordinates of each building are known. Using this *a priori* information, the approach to the mission is to pre-specify a flight plan for the vehicle that includes a camera view of all walls of all buildings within the search area (in this case, 15). The system would perform image processing on camera images looking for the identifying symbol in Fig. 1, and updating a tracker, containing a list of candidate symbol locations and orientations that is continuously updated based in processed image data. After the flight path is completed, the top symbol candidate is utilized to select the building. This building is then searched in a similar manner, this time only the walls of the selected building are searched, and the search is for openings rather than the symbol. The top candidate opening is then selected. The criteria used to select the symbol and opening are described in Sections III and IV respectively. Most flights were conducted at an altitude of 100 feet, at a relatively slow speed of 15 feet per second to allow the image processing to complete its tasks, with the camera tilt angle set to approximately 45 degrees down, and a zoom setting that places the wall of a single building to be in view vertically and a portion of the building wall in view horizontally.

II. GTMax Research UAV

This section includes a description of the GTMax research UAV hardware and software utilized for flight testing of the automated search capabilities described in sequel.

A. Hardware

The GTMax research UAV was utilized for this work, illustrated in Fig. 3, and is based on the Yamaha R-Max Industrial Helicopter airframe, which is normally utilized as a remotely piloted aircraft for crop spraying in Japan. The basic vehicle has a rotor diameter of 10.2 feet, a 246cc 21 horsepower 2-cylinder water-cooled engine, and an empty weight of approximately 125 pounds.

Fig. 3. GTMax Research UAV as configured for this work

The hardware components that make up the baseline flight avionics include general purpose processing capabilities and sensing, and add approximately 35 pounds to the basic airframe, leading to a total weight of approximately 160 pounds. The digital interface to the basic Yamaha vehicle is via a modified Yamaha Attitude Control System (YACS) interface that allows raw servo commands to be given without modification by this built-in stability augmentation system. The research avionics configuration utilized in this work includes:

- 266 MHz Pentium II Embedded PC, 500 Mb Flash Drive
- 850 MHz Pentium III Embedded PC, 2 Gb Flash Drive
- Inertial Science ISIS-IMU Inertial Measurement Unit
- NovAtel OEM-4, differential GPS
- Honeywell HMR-2300, 3-Axis magnetometer
- Custom made ultra-sonic sonar altimeter
- Custom made optical RPM sensor
- YACS: Vehicle telemetry (RPM, voltage, low fuel) and Actuator control interface
- 11 Mbps Ethernet data link and an Ethernet hub
- FreeWave 900MHz spread spectrum serial data link
- Axis 2130R pat, tilt, and zoom network camera

These components have been packaged into exchangeable modules: 2 computer modules, the GPS module, the data link module (wireless Ethernet, wireless serial, Ethernet switch), and the IMU module. These modules are placed in a vibration-isolated rack below the main body of the helicopter, shown in Fig. 4. Each module has its own power regulation, forced air-cooling, and Electro-Magnetic Interference (EMI) shielding. There is also a sonar/magnetometer assembly at the tail, a power distribution system including circuit breakers near the module rack, and mounting points for camera systems and other components under the nose.

The power distribution system utilizes an onboard generator, which outputs 12V DC. It includes a hot-swappable connection to use external power, and each component has a dedicated individual circuit breaker. External wiring for the modules consists of RS-232 Serial, Ethernet, and 12V DC only. Wiring is routed on one side of the module rack. The other side is kept free and available for temporary hookups (e.g. Ethernet), status LEDs, and switches, visible in Fig. 4. The complete wiring diagram including a typical configuration of RS-232, Ethernet, and power wiring is shown in Fig. 5; note the compartmentalization in modules and the interface to the YACS via multiple serial lines.

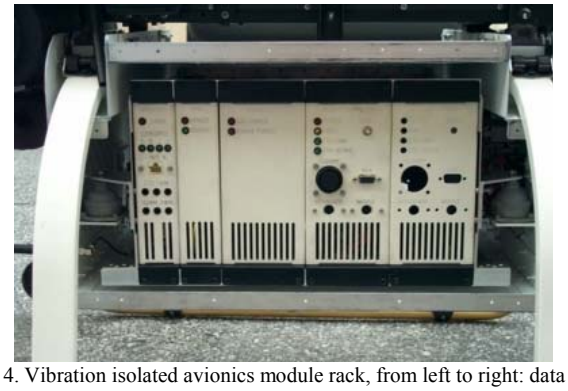

Fig. 4. Vibration isolated avionics module rack, from left to right: data link module, GPS module, IMU module, computer module #1 (flight computer), computer module #2 (auxiliary)

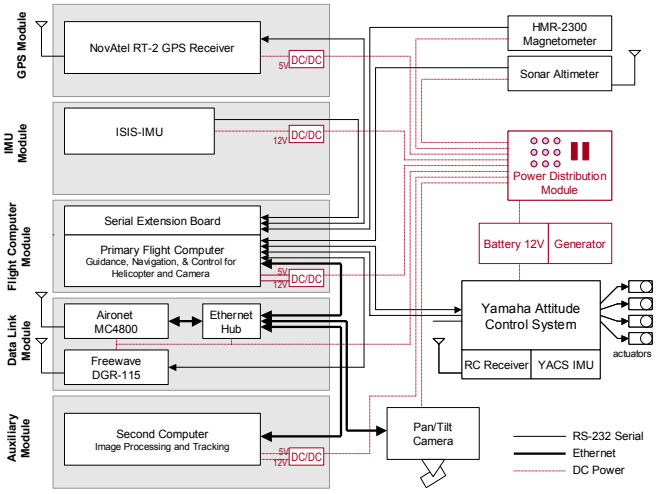

Fig. 5. GTMax wiring diagram, including separation into modules, each with their own power regulation, air-cooling, and electromagnetic interference shielding

B. *Software Architecture*

The onboard software runs on the two onboard computers, referred to as the primary flight computer and the secondary computer. The Ground Control Station (GCS) software, Fig. 6, runs on the ground on one or more laptop computers, and is used primarily for system operators to interact with the onboard systems. Simulation-specific software refers to any software that is not used in the flight configuration. All of the above software is included in the GCS or simulation-tool builds. Typically only the onboard software is included in a primary flight computer or secondary computer build in addition to any test-specific software.

Fig. 6 Ground Control Station displays, (top) real-time video from the onboard camera and processed imagery; (bottom) monitoring onboard systems with warning lights, a moving-map display, and other monitoring tools

The navigation system running on the primary flight computer is a 17 state extended Kalman filter. The states include: vehicle position, velocity, attitude (quaternion), accelerometer biases, gyro biases, and terrain height error. The system is all-attitude capable and updates at 100 Hz [5]. The flight controller is an adaptive neural network trajectory following controller with 18 neural network inputs, 5 hidden layer neurons, and 7 outputs for each of the 7 degrees of freedom [6]. These 7 degrees of freedom include the usual 6 rigid-body degrees of freedom plus a degree of freedom for rotor RPM.

The flight controller and navigation system, which coupled with the trajectory generator, is capable of automatic takeoff, landing, hover, flight speeds up to the maximum attainable by the helicopter (around 85 feet/sec) and aggressive maneuvering. The system also generates pan and tilt commands for the camera system to allow it to point at specified geographic coordinates.

The image processing and tracking components run on the second computer. During the flight, individual images come directly from the camera system in jpeg format at a rate of 1-2 Hz. They are then processed to identify either buildings or widows. A tracker combines these results along with navigation system data and camera state to update a single global picture of building and window tracking information. A mission software component utilizes this global picture to select the future flight path of the helicopter to search to the identified building and to fly to the identified building opening. This automated search subsystem, the primary emphasis of this paper, is described in detail in the following sections, along with testing results.

III. Building Identification

The McKenna MOUT site contains 15 buildings; for this mission, the correct building is identified by the symbol shown in Fig. 1, which is one meter in diameter. Traveling at the chosen velocity of 15 ft/sec, the symbol is in view for a maximum of 5 seconds. As a result, the symbol will only be visible in 5-10 of the 3000 images that are processing during the symbol-search portion of a mission attempt. Additionally, in half of these images the symbol is highly distorted due to the oblique angle at which the image is taken. These constraints make it necessary to utilize a symbol detection algorithm that minimizes the number of false rejections and utilizes a probability based tracking algorithm to make the final decision on which building is the correct one.

Fig. 7 shows a flow chart outlining the complete process for the symbol and window detection, classification and tracking. The symbol recognition system looks for line patterns in the images which are similar to the lines generated in a template of the actual symbol. Each potential symbol is filtered to determine that it only contains black and white pixels of the correct intensity. A score from the template match and color classification is then passed to the tracker. The window tracking algorithm looks at the color of each pixel before the image is processed. Then it creates a black and white image which contains only the dark objects. Then the contours around the dark objects are extracted from the image and classified based on the length of the contour. Those objects whose perimeter is an appropriate length are passed to the window tracker along with information about the windows darkness and size. The tracking algorithm is common for both the window and symbol modes. It takes the location and score of each object along with position and attitude information from the primary flight computer, and updates an array containing the position of the objects in the local geographical reference frame. Finally, it calculates a probability for each object based on how many times the object has been seen, the information from the image processor.

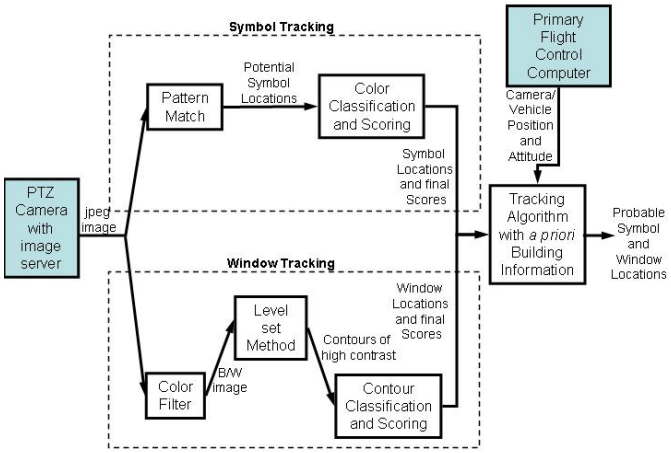

Fig. 7. Flow Chart indicating the process used to identify the building.

A. Symbol Detection

The image-processing algorithm, which was used to find the symbol, has two stages. The first stage applies a shape-matching filter to each image and then the results are filtered by color to determine the final probability of their being a symbol. By utilizing a shape match with a very low probability of false rejection followed by a stringent rejection filter, a highly accurate symbol detection algorithm is obtained.

1) Shape Matching

The shape matching is performed using Halcon, developed by MvTec Software GmBH [7], which is a commercially available library that provides C, C++, and Visual Basic routines for a number of image processing functions. In order to utilize the shape matching features available in Halcon, a template of the shape needs to be generated using the shape template generation function. The shape matching process is expedited by encoding most of the matching information inside the template. The template generation utilizes an edge detection algorithm to make line templates of an image feature at specified contrast levels. Information is collected about the template, at each contrast level, for an array of rotation angles. Then the shape match function searches the image for a specified subset of the template permutations.

Here, the template was generated using a section of the interior of the symbol taken from practice images captured with the GTMax imaging system. Character representation was used for the images; therefore, the grayscale value of each pixel is a number between 0 and 255, where 0 is black and 255 is white. The contrast levels refer to a relative change in intensity of the grayscale image. The template was created for contrast levels between 10 and 30, where the maximum is a change of 256. During the template construction process it is important not to include the outer ring of the symbol, as seen in Fig. 8. The contrast between the outer ring and the building is a function of the building itself, and it is important not to include a bias for a particular building in the filter, as it could cause an incorrect match to have a higher score than the correct match.

Fig. 8. Template selection for shape matching function.

The shape matching was performed on each image using a safe heuristic search for the template, where a safe search indicates that the chance of missing the template is minimized. The level of safety of the search is parameterized by a variable named the greediness factor; setting the greediness factor to 0 and the allowable rotation of the pattern to 360° creates the safest search. In addition, the speed and safety of the search is affected by the minimum score, which denotes the minimum score that an object must obtain before it is carefully examined, where scores range from 0 to 1. The minimum score affects the speed of the search, the number of false identifications, and the allowable distortion for the real symbol. Based on practice images, it was determined that a good value for the minimum score was 0.5. This approach to the pattern match produces a lot of incorrect matches, but also has a very low rate of false rejections, which allows the results to be further categorized with supplementary testing.

2) Color Filtering

Each symbol candidate that is returned by the shape matching function goes through an additional classification process to reduce the number of symbol matches sent to the tracker. The shape-matching function is only concerned with lines and, therefore, does not utilize any information about the color of the candidate location. Since the symbol is black and white, all symbol candidates that do not meet this criterion can be ruled out. However, it is often difficult to set up a filter that can distinguish black objects because they generally appear as gray rather than black under natural lighting conditions. When looking at a grayscale image, which is simply a weighted average of the individual color contributions, there are many color combinations that have the same intensity level as black. This issue can be circumvented, because black objects tend to have a uniform color distribution. Therefore, by first determining if the pixel contains a dominant color it is possible to make a decision about whether or not the pixel is actually black.

To classify the symbol, the color filter looks at the pixels inside a circle of set radius that is determined by the size of the template. Once again it is important to ensure that the filter only looks at pixels that are inside of the symbol in order to eliminate any preferential treatment to buildings that are certain colors. The color contributions for each pixel are then compared to determine if the difference between them is within a certain threshold, if it is not the pixel is classified as a color and given a score of zero. If the colors are judged to be uniform in composition, the grayscale value is compared to another threshold to determine if the pixel is black or white; if it is black it is assigned a score of 1 and if it is white it gets a score of 2. By first determining if a pixel has a dominant color, it is possible to use a much higher threshold between black and white; during flight testing, images of the symbol that had intense sun glare were correctly classified using a threshold between black and white of 175 out of 255, which, without the color classification, would be an unreasonably high value for black pixels. Once each pixel has been classified, the average score for the region is computed. For a black and white symbol the average score should lay between 1.0 and 2.0, where the distribution of scores is dependant on the lighting conditions.

B. Test Results for Symbol Identification

The symbol identification was tested using sequential images obtained while flying sample mission trajectories. This provided imagery suitable for testing the acquisition phase of the symbol detector as well as the rejection and scoring phases. The test arrangement utilized a pure-software simulator configuration. In the test configuration, an output of the intermediate results from the color filter is produced. This allows one to effectively tune the nominal conditions in the filter.

There are four possible output cases from the symbol detector: a false rejection, a false identification, correct rejection and a correct identification. The first case was minimized by utilizing a safe search for the pattern match and relaxing the pattern matching constraints until the number of false rejections was at a tolerable level. The second case is minimized using the color filter. Fig. 9 shows a case where the color filter rejects the symbol location because it contains too much color.

Fig. 9. Rejection of a symbol candidate for excessive color on the lower-right window

Once it has been determined that a candidate symbol location contains mostly black and white, the ratio of black to white pixels is checked to determine if it meets the guidelines for the anticipated pixel combination for the symbol in the

given lighting conditions. Fig. 10 depicts the case where the color filter recognizes the area as black and white, but rejects it for being too dark.

Fig. 10. Rejection of a symbol candidate for excessive darkness, on the upper center window.

The final case is depicted in Fig. 11, a valid identification requires that all of the conditions of the color filter are met. In this case the location is assigned a score, which reflects how well it matches the pattern and how close it is to the nominal symbol score in the color filter.

Fig. 11. Valid acceptance of a symbol candidate.

C. *Symbol Tracking*

The symbol tracker takes the location of a valid symbol candidate on each individual image, and transforms it into local geographical coordinates (LGC). Each symbol track is then evaluated to determine if it is a subsequent measurement of a pre-existing track or a new symbol-candidate track altogether. If it is a subsequent measurement, the new information is combined with the current information to obtain the best estimate of the actual symbol-candidate location using a Bayesian tracking technique. Then, once all of the buildings have been searched, the location with the highest probability is taken as the symbol location. Once the symbol location is determined in the local frame, the building closest to the symbol, which satisfies all of the heading constraints embedded in the symbol object, is determined. This is chosen as the correct building for the final portion of the mission.

The first task of the tracker is to convert all of the relevant symbol information from the two dimensional image coordinates into LGC. This is done by first projecting the symbol image coordinates onto a reference plane, which is parallel to the ground at the expected height of the symbol. In this case, it was decided to project the symbol onto a plane six feet above the ground, which would lead to relatively little position error for any actual symbol height in the village. The first step in this process was to find a vector from the camera towards the direction of the symbol in the image, in the camera coordinate system, which is shown in Fig. 12. This is done by transforming the offset of the symbol in the image into a portion of the field of view using a small angle approximation. Then, by utilizing an image projection plane which is perpendicular to the camera and 1 unit away from the camera in the x-direction, it is possible to define a vector, $\vec{r}_C$, from the camera to the projection of the symbol location on the projection plane.

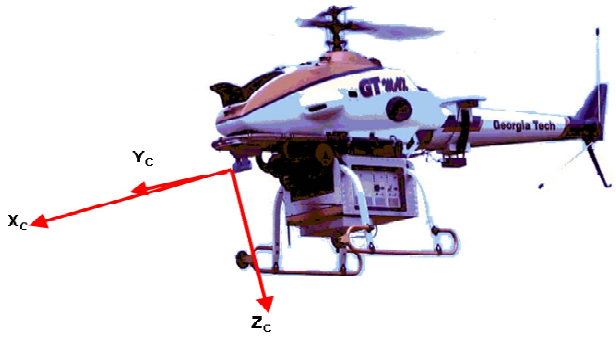

Fig. 12. Camera Coordinate Frame, X_C goes out the center of the camera view

$$\begin{aligned} r_{CX} &= 1 \\ r_{CY} &= \tan\left(0.5 \cdot FOV_X \cdot px\right) \\ r_{CZ} &= \tan\left(0.5 \cdot FOV_Y \cdot py\right) \end{aligned} \quad (1)$$

Where FOV is the field of view of the camera in the x_{IM} or y_{IM} directions and px and py are numbers between -1 and 1 which indicate the position of the symbol-candidate measurement in the image in image coordinates. Once $\vec{r}_C$ is determined it is normalized to a unit vector and transformed

into the LGC using the direction cosine matrix (DCM) and vehicle state, $\hat{v}_L$, from the time the image was captured (data computed by the navigation system on the primary flight computer), this results in the unit vector, $\hat{r}_L$. Then the direction of the object was evaluated to ensure that it is a valid direction, i.e. within the search area. If it was not, then the result is thrown out. The final step of the conversion is to calculate the intersection of the vector with the reference plane at height ref_Z.

$$p_{LX} = v_{LX} - \frac{v_{LZ} - ref_Z}{\hat{r}_{LZ}} \hat{r}_{LX}$$

$$p_{LY} = v_{LY} - \frac{v_{LZ} - ref_Z}{\hat{r}_{LZ}} \hat{r}_{LY}$$

$$p_{LZ} = ref_Z \tag{2}$$

The result of this formulation is not the exact location of the symbol but rather a projection of its location onto the reference plane with respect to the vehicle location. This means that the resulting symbol position for sequential symbol hits, which are viewed from different angles, will not be the same; this issue as well as the error in the position estimation for the helicopter itself is dealt with in the tracking mechanism and the process which makes the final decision about the correct building, described below.

Once the location of a new symbol track is found, a probability that the track is the correct symbol is assigned. Each new symbol-candidate sighting is given an initial nominal probability of 0.1, and then additional comparison tests are used to modify the probability of the track by (3).

$$P(k) = P(k-1) + (1 - P(k-1)) \cdot S \cdot P_S \tag{3}$$

where k is the comparison number and S is the score between 0 and 1 which is obtained from the pattern matching algorithm, and P_S is a scaling factor to control the growth rate of the probability factor with additional comparisons. Once the final probability is determined it is used to calculate a weighted initial position variance for the symbol-candidate measurement. This variance is based on the estimated error in position and angular measurements $\sigma_p^2, \sigma_\theta^2$ from the navigation system, and formulated as:

$$\sigma^2 = \frac{\sigma_p^2 + \sigma_\theta^2 \cdot R}{P} \tag{4}$$

where R is the estimated range to the symbol-candidate, and P is the probability found from (3).

Once the characteristic data for the location is found, it is examined to decide if it is a new object or one that is already being tracked. This decision is made based on the distance between the new symbol location measurement and all existing symbol track locations as well as the heading at which the new object and the old object were detected. The first condition ensures that objects which are considered to be the same are close enough together, for symbol detection this threshold is kept reasonably large in order to ensure that symbol tracks observed at slightly different headings are recognized as a single symbol track. This does not create problems for tracking symbols since the rate of occurrence for new tracks is low. The second condition ensures that a new object is not combined with a nearby track that is not in view of the camera. If it is decided that the proposed symbol location is a new symbol track, the position, heading and variance information are stored. If it is a new instance of a symbol that is already being tracked, the data from the new and old tracks are fused together using a weighted average based on the position variances. The variance for the newly update track is given by:

$$\sigma_t^2 = \frac{\sigma_t^2 \sigma_{nt}^2}{\sigma_t^2 + \sigma_{nt}^2} \tag{5}$$

where subscript t denotes the existing track and subscript nt denotes the new measurement. Similarly, the position and heading information can be updated using:

$$Y_t = \frac{Y_{nt}\sigma_t^2 + Y_t \sigma_{nt}^2}{\sigma_t^2 + \sigma_{nt}^2} \tag{6}$$

where Y is the variable being updated.

IV. Window Mapping

A. Window Detection

The window/door opening detection utilizes the *geometric active contour* or *snake* method [8,9,10,11,12] to finding contours around dark objects that have high contrast edges. One of the most difficult challenges to overcome in image processing is noise in the image. Since the window detector is looking for naturally dark objects it is possible to use a color filter similar to the one used in the symbol detection to overcome this issue. Then by using a fast-marching level set method the processing time for each image is decreased, which increases the number of window measurements provided at a given search flight speed.

1) Image Filtering

In flight images contain shadows, clouds, grass, and many more objects that could potentially produce a contour. In order to minimize the number of extraneous contours in an image, a color filter is employed. The process is similar to the filter utilized in the symbol detection. Since the open windows and doors are naturally dark, they exhibit properties similar to black objects. Therefore, by eliminating colored pixels from the image, it is possible to construct a black and white image that contains only the dark objects.

The first step in the filter is to look at each pixel and determine if it has a dominant color, as was done in the color filter for the symbol. If it does have a dominant color then it is set to white. If it does not have a dominant color, then the grayscale value of the pixel is evaluated to determine if it is black or white; for window detection this threshold is set to 90 out of 255. The result of this filter is a black and white image, which only contains dark features. Using the filtered image in the subsequent processes helps the contour detection by increasing the contrast for window edges and by eliminating much of the noise in the image.

2) Geometric Active Contours

a) Background on Active Contours

Geometric active contours or *snakes* were used in the window tracking algorithms. Accordingly, this theory is briefly reviewed here. Snakes are autonomous processes that employ image coherence in order to track various features of interest over time. They fit very naturally into a control framework and indeed have been employed in conjunction with Kalman filtering; see [9] and the references therein.

In particular, deformable contours have the ability to conform to various object shapes and motions. Snakes have been used for segmentation, edge detection, shape modeling, and visual tracking. In the classical theory of deformable contours, energy minimization methods are used; where controlled continuity splines are allowed to move under the influence of external image dependent forces, internal forces, and certain constraints set by the user.

As is well-known there may be a number of problems associated with this approach such as initializations, existence of multiple minima, and the selection of the elasticity parameters. Moreover, natural criteria for the splitting and merging of contours (or for the treatment of multiple contours for tracking of multiple objects) are not readily available in this framework.

In [10,11,12] an active contour model is proposed to help treat such problems, and so make it very useful for tracking. The underlying mathematics is based on the Euclidean curve shortening evolution which defines the gradient direction in which a given curve is shrinking as fast as possible relative to Euclidean arc-length [13], and on the theory of conformal metrics. The Euclidean arc-length is multiplied by a conformal factor defined by the features of interest, and then the corresponding gradient evolution equations are computed. Therefore, the features to be capture lie at the bottom of a potential well to which the initial contour will flow. Moreover, this model may be easily extended to extract 3D contours based on motion by mean curvature [8,10].

b) Geometric Active Contours

As alluded to above, the approach used is based on an active contour model founded upon the calculus of variations and length/area minimizing ideas. Full details of the theory may be found in [10,11,12].

Let $C = C(p,t)$ be a smooth family of closed curves where t parameterizes the family and p the given curve. The geodesic active contours model is formulated from [10,11]. The basic idea is to change the ordinary Euclidean arc-length function by multiplying by a conformal factor ϕ which is assumed to be a positive, differentiable function. The resulting **conformal Euclidean metric** is given by multiplying the ordinary Euclidean metric by the conformal factor. As in ordinary curve shortening [13], the corresponding gradient flow for the modified length functional is:

$$L_\phi(t) = \int_0^L \phi \, dv \tag{7}$$

where dv is the ordinary Euclidean arc-length. Taking the variational derivative and integrating by parts, the gradient flow is found to be:

$$\frac{\partial C}{\partial t} = (\phi\kappa - \nabla\phi \cdot N)N \tag{8}$$

where κ, N denote the curvature, (inward) unit normal, respectively. As long as the flow remains regular, there will be convergence to a closed geodesic in the plane relative to the conformal Euclidean metric. Regularity may be deduced from the classical curve shortening case. Finally, if one wants to put in an area-minimizing term [12], then the complete equation becomes:

$$\frac{\partial C}{\partial t} = (\phi(\kappa + \nu) - \nabla\phi \cdot N)N \tag{9}$$

In the latter equation, $\nu > 0$ is called the *inflationary parameter*, and is user-determined.

In the context of image processing, ϕ is a stopping term that depends on the image. The stopping function that was implemented for this problem as well as the numerical implementation of (9) is described below.

c) Numerical Implementation

In the tracking application described in this paper, it was essential to make an algorithm fast enough to be used in real-time. Therefore, a simplified version of (9) is used, where the curvature term is neglected and ν is set to 1. This corresponds exactly to the area-minimizing flow [12], and can be very quickly implemented using the fast-marching method as described in [14]. The function $\phi(x,y)$ is constructed to select high contrast edges. Accordingly,

$$\phi(x,y) = \left(\frac{1}{1 + \|\nabla I\|^2} \right) \tag{10}$$

was chosen where I(x,y) is the intensity of the image.

The process is initialized by defining the initial surface C(p,0), as shown in Fig. 13. The initial form of the family of curves C is chosen to be circular. Then the initial contour is the defined as the points of contact between the surface and the 2-D image. After the initialization the contour can be copied to every frame, therefore, it only needs to be made once.

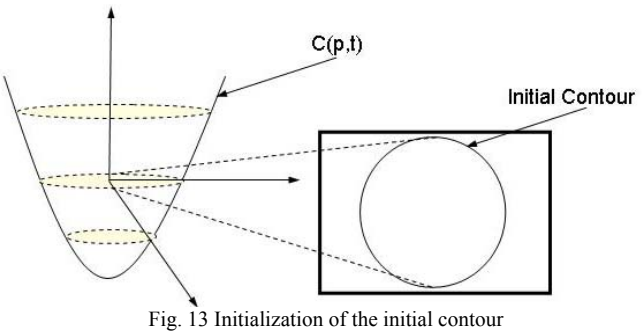

Fig. 13 Initialization of the initial contour

To make a faster algorithm the smoothness of the evolving curve is ignored. The curve moves with constant speed, and the evolution is stopped when the stopping term is sufficiently "small" (defined by a predefined thresholding value). Therefore, it is possible to eliminate the higher order derivative requirements needed for the curve evolution. If the function $\phi(x, y)$ satisfies the stopping condition, the curve is frozen in a neighborhood around the given pixel; this preserves the edges during the evolution. A large neighborhood produces a more robust result but can sacrifice accuracy. Through experimentation, it was determined that a 5x5 neighborhood presented the best trade-off. These settings dramatically speed up the processing, allowing speeds of 5 fps for a 200x200 pixel image to be obtained on a 1700 MHz Pentium 4 computer.

The process is demonstrated in Fig. 14, where the evolving contour moves with constant speed and stops when it satisfies the stopping condition. Inside the object, the evolving curve does not pass zero except in the circular and rectangular regions because of the intensity condition in the definition of $\phi(x, y)$. Since the curves can be oriented, one can tell which points are inside and which are outside which allows the gray-level image to be converted into a binary one.

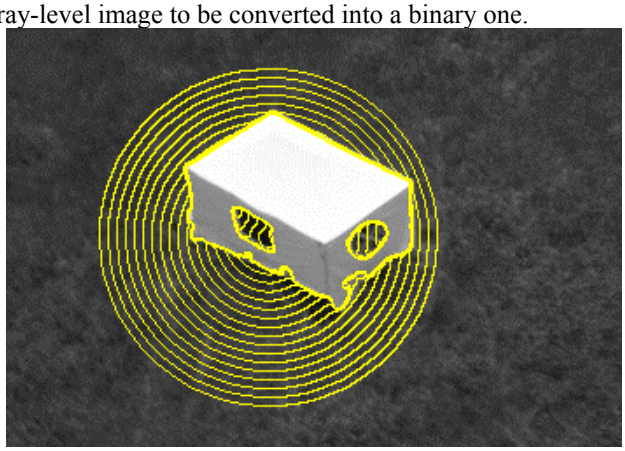

Fig. 14. Geometric Active Contour Method: the image shows the convergence steps in locating the three contours, two around the openings and the third around the building itself

3) Window Contour Extraction and Characterization

The geometric active contour method is used to get multiple contours from each image. The contours are broken into individual features, and then geometric characteristics are extracted from the individual contours, which allow the tracker to determine which objects are most likely windows. The contours are extracted from the line image by systematically searching for lines and then removing the associated contour from the image. This is done using a left-handed removal technique. In this technique, the region around the current pixel is examined to determine in which direction the contour continues. In the event that it continues in more than one direction, the left most contour pixel is followed. The extracted contours are then filtered in post-processing based on their characteristics. The first characteristic examined is whether or not the contour is closed. Although most of contours are closed, if an object lies on the boundary of the initial contour the entire contour might not be captured; therefore, leaving it open. For this application, these contours were immediately rejected as candidates; however, they need to be completely removed from the line image before the next contour can be found. Since contour removal can begin anywhere along the contour, if it is not closed the left-hand removal technique can be halted before the entire contour has been removed. In this case, the rest of the contour needs to be removed by a right-hand removal technique before the process can continue.

Once a contour has been classified as closed it is filtered for length, which helps speed up the post-processing. Since, valid window contours will have relatively small-range of possible lengths; contours that are far longer or shorter than expected are ruled out. This also helps increase the accuracy of the tracker by filtering out ill-formed contours that do correspond to real window locations, which regularly occur in real images due to noise. If the contour passes these two preliminary filters, the more computationally expensive geometric characteristics are calculated and the information is passed to the tracker for classification.

For each window, four sets of geometric data are sent to the tracker: the center of mass, the interior area, the corner locations, and the darkness of the interior. The center of mass is found by averaging the pixel locations over the interior area. Once this is determined, the location of the corners must be found. Determining corners on a smooth contour can be an intricate task. However, it is reasonable to expect all of the windows and doors to be square and oriented nearly vertically in the image; therefore, if the origin is shifted to the center of mass of the contour it is reasonable to expect that one corner will be located in each quadrant. With this assumption, the corners can be characterized as the farthest point from the center of mass in each quadrant. Once the corner locations have been defined, it is possible to determine a characteristic rectangle for the contour, where the width is the average width between corners and the height is the average height between corners. Then the area of the contour can be calculated from the quadrilateral, and the darkness is determined by averaging the grayscale values of each pixel inside of it.

B. Test Results

The window algorithm was tested using the same practice images as were used for the symbol. This tested the ability of the filter to eliminate noise in images that did not contain windows as well as for the entire algorithm to work on images that did contain windows. Fig. 15 shows the results of the image filter with the centroid and corner results for the windows that were found using the geometric active contours. Since the initial contour is a circular region, only a portion of the image can be evaluated at one time, causing the system to miss the openings on the upper floor.

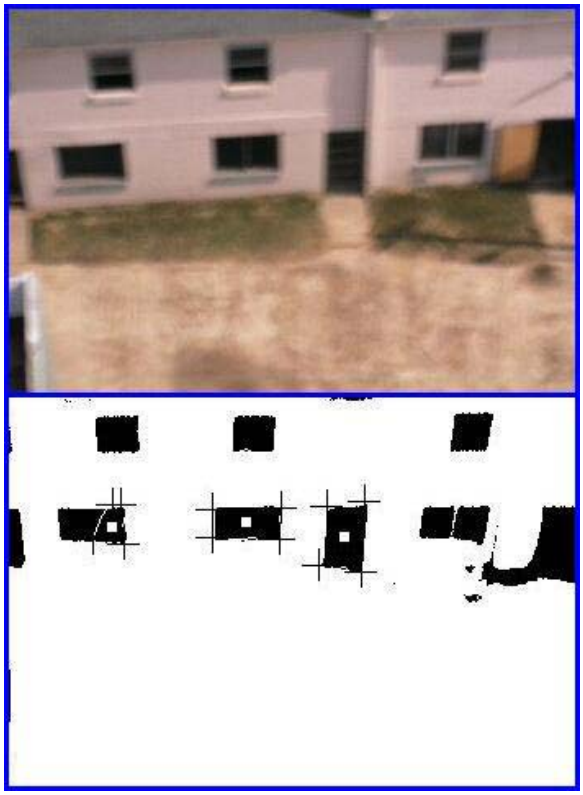

Fig. 15. Identification of the geometric characteristics of the contours.

C. *Window Tracking*

Window tracking requires a more sophisticated algorithm than the symbol tracking. Since each building has multiple stories, it is important to capture the height of the window as well as its geographical (horizontal) location. Secondly, since the windows are close to each other, their positions must be determined with greater precision in order to ensure that measurements are not incorrectly lumped together the in tracking. Since the size of the windows is unknown, there are only two measurements readily available from the image; therefore, it is difficult to determine an objects position in 3-dimensional space without some *a priori* knowledge of the location. In this case, approximate building locations are already known based on public surveys of the McKenna MOUT site, including satellite photography. Therefore, the absolute position of the window can be determined by mapping it onto the visible wall. The walls of each building are defined by choosing a characteristic corner and finding a unit vector, $\hat{w}_L$ along the wall from that corner; then the location of the corner, and the length of the wall are recorded.

The first step is to find a vector, $\hat{r}_L$, from the camera to the centroid of the window in the local frame using the steps described in section III.C; once again, checking to make sure the vector is pointing towards the ground. The building that contains the windows is known, since the mission planner directed the UAV to the building in order to begin this portion of the mission. Therefore, the correct wall can be determined by finding the angle between the unit vector $\hat{r}_L$ and $\hat{w}_L$. Fig. 16 is a diagram showing the variables used to determine the position of the window.

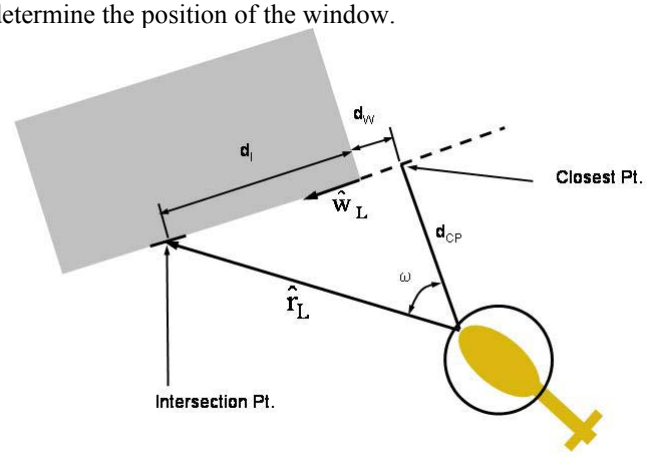

Fig. 16. Geometry for determining the window position.

The angle ω can be found by:

$$\omega = \operatorname{atan}\left(\frac{\hat{r}_{LX}\hat{w}_{LX} + \hat{r}_{LY}\hat{w}_{LY}}{-\hat{r}_{LX}\hat{w}_{LY} + \hat{r}_{LY}\hat{w}_{LX}}\right) \quad (11)$$

where the subscript X denotes the x component of the vector and Y denotes the y component of the vector. Therefore, the numerator is the 2-dimensional projection of $\hat{r}_L$ onto $\hat{w}_L$ and the denominator is the projection of $\hat{r}_L$ onto a unit vector perpendicular to $\hat{w}_L$. The window will be located on the wall that is facing the camera; therefore, |ω| should be less than 90° for the correct wall.

The closest point on the wall to the camera, (x_{CP},y_{CP}), can be located using least squares. First, the distance from the characteristic corner of the wall to the closest point is found:

$$d_W = (x_{cam} - x_{wall})\hat{w}_{LX} + (y_{cam} - y_{wall})\hat{w}_{LY} \quad (12)$$

where x_{cam} and y_{cam} are the x and y coordinates of the camera in the LGC and x_{wall} and y_{wall} are the x and y coordinates of the characteristic corner of the wall in LGC. Then x_{CP} and y_{CP} are:

$$x_{CP} = d_W\,\hat{w}_{LX} + x_{wall}$$
$$y_{CP} = d_W\,\hat{w}_{LY} + y_{wall} \quad (13)$$

Therefore, the distance from the characteristic corner of the wall to the intersection point is:

$$d_I = d_W + d_{CP}\tan(\omega) \quad (14)$$

where d_{CP} is the distance from the camera to (x_{CP},y_{CP}). Then the intersection point can be calculated as:

$$x_I = d_I\hat{w}_{LX} + x_{wall}$$
$$y_I = d_I\,\hat{w}_{LY} + y_{wall} \quad (15)$$

Subsequently, the height of the window can be determined by:

$$z_I = x_{cam} - d_{cam}\tan(\theta_c) \quad (16)$$

where d_{cam} is the distance from the camera to the intersection point in the x-y plane, and θ_C is the pitch angle of the camera.

The intersection point is then examined to see if it lies between the corners of the wall and at a height that is

legitimate for the building. Since errors in the location of the building or in the position estimate of the vehicle will result in an error in height estimate for the window, the height requirements included a margin of error above and below the building of 10 ft.

V. Full-System Results

This automated system was designed to run in real-time, meaning that it collects and processes images while flying the mission. The algorithms must work quickly to accomplish this. The camera onboard the GTMax was capable of delivering images at 5Hz. Once the image had been obtained the symbol and window detection could begin. These processes ran at different rates. There was very little extra analysis performed on potential symbol candidates, consequently the symbol detection took approximately 0.25 seconds to completely process each image on an 850 MHz computer that was running at 33% power. Conversely, the window detection and classification took approximately 1 second to completely process each image on the same computer.

The system described was tested in several ways. First, simulated images were presented to the image processing from a scene generator in the simulator. In these tests, the image-processing job is considerably easier, given the clean characteristics of these synthetic images. However, this is very good test of the object tracking and flight planning components. Three tests were performed searching all 15 buildings of the simulated McKenna MOUT village, and the correct building was selected and a valid opening was found.

Next, tests were conducted at the actual McKenna MOUT village. The initial flight tests were done with a search area including three of the buildings. On all of these tests, the system selected the correct building. In each case, the symbol was placed on a different building. Following this, four attempts were made to search the entire village of 15 buildings. On the first, the actual symbol was missed (missed detection); thrown out because it was considered too bright, presumably due to bright sunlight directly on the symbol unlike all previous tests. The nominal distribution range for the symbol score was changed to account for increased brightness. On all three subsequent tests the correct building was selected based on the symbol, appearing for approximately 5 seconds over a total search flight time of approximately 15 minutes.

A typical result for the opening search on a single building is shown in Fig. 17. This plot shows a representation of the building with the ten best opening locations from the tracker superimposed. From Fig. 17, it can be seen that the openings are correctly mapped onto the walls of the building, and these positions match actual opening locations accurately. In this case, the aircraft circled the building in a clockwise manner starting at on the northern face. The majority of the openings picked are located on the eastern side; this is possibly due to favorable lighting conditions. The height distribution shows that the system favored the openings on the bottom floor over those on the top, which is consistent with the results of the images shown in Fig. 15. As was previously mentioned, the majority of the estimation errors in both the vehicle and opening positions will manifest as errors in the height estimate. However, the height distribution is remarkably consistent. There are two anomalous readings, where the height estimate for an opening is near or below ground level; these anomalous heights correspond to the two inner most openings on the south side of the building. This is probably due to the image processor erroneously choosing objects located on the ground at the base of the building as potential openings.

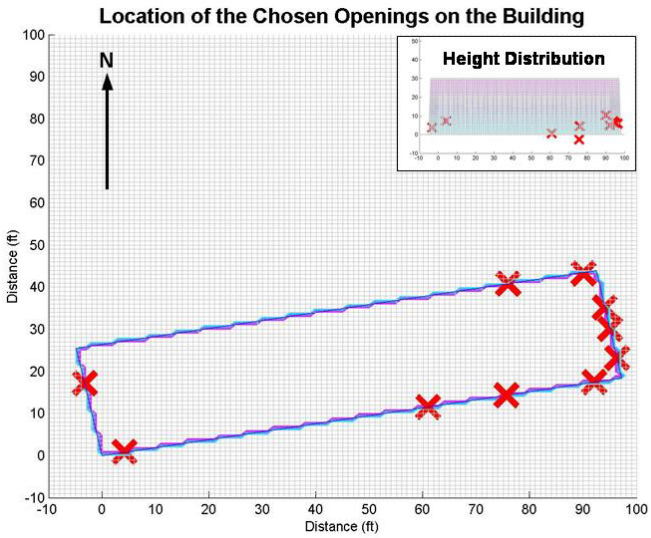

Fig. 17. Results of one flight which shows the location of the openings mapped onto the building

After searching the selected building for an opening, the UAV would hover in front of the selected opening, pointing the camera at the opening. Fig. 18 shows a typical onboard image hovering in front of the opening, in this case an open door.

Fig. 18. Camera placed with selected building opening in center, in this case an open door

VI. Conclusions

This paper described the design, development, and testing of an Unmanned Aerial Vehicle (UAV) with automated

capabilities: searching a prescribed area, identifying a specific building within that area based on a small sign located on one wall, and identifying an opening into that building. It was important to use fast image processing and tracking algorithms to ensure a rapid search. The resulting system was able to search a 15-building village automatically with speed comparable to a human operator searching on foot or with a conventional remotely piloted vehicle. Future surveillance UAV systems would ideally make use of both capabilities, utilizing the strengths of human intelligence and the human eye as well as the kinds of image processing and tracking automation described herein.

REFERENCES

[1] Proctor, A.A., Kannan, S.K., Raabe, C., Christophersen, H.B., and Johnson, E.N., "Development of an Autonomous Aerial Reconnaissance System at Georgia Tech," *Proceedings of the Association for Unmanned Vehicle Systems International Unmanned Systems Symposium & Exhibition*, 2003.

[2] Dooley, J., Taraloba, E., Gabbur, P., Spriggs, T., "Development of an Autonomous Aerial Reconnaissance System**",** *Proceedings of the Association for Unmanned Vehicle Systems International Unmanned Systems Symposium & Exhibition*, 2003.

[3] Burleson, W.L., Keller, K., Harrison, G., Corliss, F., "Southern Polytechnic State University Autonomous Remote Reconnaissance System", *Proceedings of the Association for Unmanned Vehicle Systems International Unmanned Systems Symposium & Exhibition*, 2003.

[4] Johnson, E.N. and Schrage, D.P., "The Georgia Tech Unmanned Aerial Research Vehicle: GTMax," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2003.

[5] Dittrich, J.S. and Johnson, E.N., "Multi-Sensor Navigation System for an Autonomous Helicopter," *Proceedings of the 21st Digital Avionics Systems Conference*, 2002.

[6] Johnson, E.N. and Kannan, S.K., "Adaptive Flight Control for an Autonomous Unmanned Helicopter," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2002.

[7] MVTec (2003, May 23) Halcon/C++ Reference Manual, MVTec Software GmbH, München, Germany. [Online]. Available: http://www.mvtec.com

[8] A. Tannenbaum, "Three Snippets of Curve Evolution Theory in Computer Vision, " *Mathematical and Computer Modeling,* vol. 24, pp. 103-119, 1996.

[9] R. Malladi, A. Sethian, and B. Vemuri, "Shape Modeling with Front propagation : A Level Set Approach", *IEEE Transactions on Pattern Analysis*, (1995)

[10] S. Kichenassamy, A. Kumar, P. J. Olver, A. Tannenbaum, and A. Yezzi, ``Conformal curvature flows: From phase transitions to active vision," Archive for Rational Mechanics and Analysis, vol. 134, pp. 275-301, 1996.

[11] V. Caselles, R. Kimmel, and G. Sapiro, `` Geodesic active contours," *International Journal of Computer Vision*, 1997.

[12] Y. Lauziere, K. Siddiqi, A. Tannenbaum, and S. Zucker, ``Area and length minimizing flows for segmentation," *IEEE Trans. Image Processing*, vol.7 , pp. 433-444, 1998.

[13] M. Grayson, ``The heat equation shrinks embedded plane curves to round points," *J. Differential Geometry*, vol. 26, pp. 285-314, 1987.

[14] J. A. Sethian*, Level Set and Fast Marching Methods*, Cambridge University Press, 1999.